



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/992,130	11/14/2001	Ronald Hilton	AMDH-08152US0 DEL	4626
21603 7590 01/12/2007 DAVID E. LOVEJOY, REG. NO. 22,748 102 REED RANCH ROAD TIBURON, CA 94920-2025			EXAMINER SAXENA, AKASH	
			ART UNIT	PAPER NUMBER
			2128	
SHORTENED STATUTORY PERIOD OF RESPONSE		MAIL DATE	DELIVERY MODE	
3 MONTHS		01/12/2007	PAPER	

**Please find below and/or attached an Office communication concerning this application or proceeding.**

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

**Office Action Summary**

Application No.

09/992,130

Applicant(s)

HILTON, RONALD

Examiner

Akash Saxena

Art Unit

2128

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 19 October 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-3 and 5-27 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-3 and 5-27 is/are rejected.
- 7) ☒ Claim(s) 10,24 and 26 is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- ☒ Notice of References Cited (PTO-892)
- ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- ☐ Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_
- ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_
- ☐ Notice of Informal Patent Application
- ☐ Other: \_\_\_\_\_

### **DETAILED ACTION**

1. Claim(s) 1-3 and 5-27 has/have been presented for examination based on amendment filed on 19<sup>th</sup> October 2006.
2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 19<sup>th</sup> October 2006 has been entered.
3. No Claim(s) is/are amended/cancelled.
4. No Claim(s) is/are added with this amendment.
5. Claim(s) 1-3 and 5-27 are rejected under 35 USC § 112.
6. Claim(s) 1-3, 5-9, 11-23, 25 and 27 remain rejected under 35 USC § 103.
7. Claims 10, 24 and 26 are objected to being dependent from rejected claims and may be allowable if presented in independent format after curing the deficiencies of the independent claims.
8. The arguments submitted by the applicant have been fully considered. Claims 1-9, 11-23, 25 and 27 remain rejected and this action is made NON-FINAL. The examiner's response is as follows.

***Response to Applicant's Arguments***

9. Response to Section 11.2.1

*Regarding Section 11.2.1.1:*

*-- No argument presented other than stating the examiner's rejection.*

*Regarding Section 11.2.1.2-11.2.1.3:*

Examiner admits that feature "indexing table" is recited in the claim language and withdraws the argument that the "indexing table" feature is not claimed.

10. Response to Section 11.2.2

Applicant seems to be arguing that applicant's indexing table represents species and the TLB (translation look-aside buffer) taught by the prior art represents genus.

Examiner had presented the argument that TLB is the species instead, which was acknowledged by applicant in response first office action as being more complex than an indexing table (genus). Examiner maintains the rejection as stated in MPEP 2131.02 where the species anticipates a genus.

(Response A: Section 10.2 by applicant states in part) ... For example the Examiner must agree that the indexing table of the applicant could not be used to replace the TLB of the smith'1982 publication. In a similar way, why would anyone adopt the complex structure of TLB, with the large amounts of data implied, to a simple task of a small table in applicant's invention. ..."

*Regarding Section 11.2.2.1-11.2.2.2:*

*-- No argument presented other than stating the examiner's previous rejection and alleging that TLB cannot perform the function of indexing table.*

*Regarding Section 11.2.2.3-11.2.2.4:*

Applicant has presented 'A' and 'B' (stated below again) related to action performed when the indication indicates if a particular block is translated or otherwise.

Art Unit: 2128

(A) "if the indication indicates that the said particular block has not been translated, going to step of executing translated instructions."

(B) "if the indication indicates that the said particular block has been translated, checking the said translation store to determine if the legacy instruction data has been modified"

The argument being presented is indexing table goes directly to executing translated instruction if (A) and in TLB the checking (for A it seems) must occur thereby losing efficiency. Examiner is confused by applicant's argument; it seems that applicant is implying that there is no checking done in indexing table for indication to indicate if the translation is performed.

On the contrary the language of the claim clearly states that the "indication indicates...", meaning some process performs a looked up (checking) in the indexing table for the indication to indicate the status of translation.

As for the maximizing the performance & efficiency, applicant seems to be arguing features that are not clearly cited in the claim as to how the performance and efficiency is increased. No evidence is presented to this effect.

*Regarding Section 11.2.2.5-11.2.2.7:*

Applicant with the use of drawings has alleged that TLB maximizes the YES path and indexing table is maximizing the NO path. Further argument is made that the prior art does not indicate maximizing the NO path.

In response, first, there is no limitation presented in the claim that indicates maximizing the YES path and NO path or the intended difference in performance & efficiency of the indexing table over TLB. Secondly, the arguments presented although detailed, are weak, as there is no factual information provided that TLB does not maximize the NO path. Examiner requests applicant to indicate where in

Art Unit: 2128

the claim language the emphasis in maximization of path, performance & efficiency is claimed. Further, TLB meets the function performed indexing table to provide fast and efficient practical implementation of indexing.

Applicant's arguments are found to be unpersuasive.

#### 11. Response to Section 11.2.3

##### *Regarding Section 11.2.3.1*

Applicant is arguing that Mann'295 is does not disclose reuse of the code by other block table entries (individual instructions) as asserted by examiner.

Examiner respectfully disagrees, and refers applicant to Fig.3. The Tag E in the block table entry indicates alternate entry point into the block of code (Mann'295: Col.6 Lines 47-61). If we follow the arrow, we can see that Tag E goes to *second* Table 80 instead of *first* Table 80, thereby indicating that host code 88 is not sequentially executed. Further, arrow 84 from second Table 80 points back to first Table 80 and first Table 80 only starts the host code 88 back again, as indicated by arrow 82. Similar execution pattern can also be followed from first Table 80 to second Table 80 via arrow 85. This clearly shows reuse of the "host code 88" by other "block table entries" or instructions (See elements 72, 74, 76 as features of other block table entries 70) as asserted by examiner. This also shows many (block table entries) to one (host code) relationship.

Further the question is whether translated host code is reused, not if entries belong to same block table or different block tables (block tables demarcated by "0" Tag 72 bit). The reason why it is irrelevant if entries come from different block tables is

because the division of original code into block tables is not instruction type dependent and any instruction can be the first instruction (Tag 72 = F) of the block table (Mann'295: Col.6 Lines 13-15).

To reassert, the important point is that host code 88 is reused by many block table entries 70 (instructions). In any practical implementation of instruction reuse same data would not be executed (i.e. instruction reuse does not imply data executed is also reused, otherwise the exercise may be futile – running same instruction with same data again and again). This is also precisely the problem not addressed by Mann'295 where there is no distinction made between the instruction and the data in the host code 88, at least in the disclosure.

To make and use the teaching of Mann'295, a Table 80 has to be implemented. TLB (Translation Lookaside Buffer) are practical implementation of the Table 80 which maps the virtual address of the block table entries 70 to real address in host code 88. Smith'1982 discloses the problem of "synonym" as multiple block table entries 70 mapping to same host code 88 (Smith'1982: Pg. 510 section 2.9).

Applicant argues that examiner finds the problem being solved in Mann'295 which is not based on the operations presented in Mann'295. Any practical disclosure or teaching, does not discredit itself, in fact it shows the deficiencies in the prior art. Hence the deficiency in Mann'295 would not be apparent from Mann'295. Mann'295 discloses and claims a "methodology" for code emulation, not the exact details of the implementation (e.g. implementation of Table 80). If one would implement the methodology, for example a Table 80 in Fig. 3, an exemplary practical

implementation of such a table through TLB, it would bring in implementation issues, which are not limited by which table structure is used (TLB, indexing table etc).

Hence to make and use the teaching of Mann'295 considerations presented in Smith'1986 must be taken into account.

Applicant alleges that the reason Mann'295 would not face the issues disclosed in Smith'1986 is operation of Mann'295 are not as suggested by examiner. Examiner asserts that such a manner is not only asserted by examiner, but it is also disclosed by Mann'295 as explained above. Hence combination of Mann'295 with Smith'1986 would be obvious to one skilled in the art of code emulation.

*Regarding Section 11.2.3.2:*

Applicant summarizes the arguments presented, response to which is addressed above. Examiner finds applicant's argument unpersuasive.

#### 12. Response to Section 11.2.4

*Regarding Section 11.2.4.1-11.2.4.2:*

Applicant has argued that Mann'295 is wasteful of processing time because regardless if data is changed, when the self modifying code is reached, execution is aborted and transfers back to interpreter. Applicant asserts in instant application such a branch is not made unless the data is changed.

Examiner would like to first point out that "self modifying code" is an art specific term that is not claimed in claim 1. Although applicant's argument may be valid, claim does not reflect that.



Art Unit: 2128

Wikipedia online dictionary defines Self-Modifying Code as:

In computer science, self-modifying code is code that alters its own instructions, whether or not it is on purpose, while it is executing.

The emphasis is on instructions, which is further explained as “opcode” or operator being modified during execution.

For example:

Pseudocode example:

```
repeat N times {  
  if STATE is 1  
    increase A by one  
  else  
    decrease A by one  
  do something with A  
}
```

Self-modifying code in this case would simply be a matter of rewriting the loop like this:

```
repeat N times {  
  
  increase A by one  
  do something with A  
}
```

```
when STATE has to switch {  
  replace the opcode "increase" above with the opcode to decrease  
}
```

Claim 1 for example neither claims “self-modifying code” specifically, nor generically as “modifying opcode/operator” during instruction execution.

For example claim 1 recites:

“ if the indication indicates that said particular block has been translated, checking the said translation store to determine if the legacy instruction data **[emphasis added]** has been modified and if modified, repeating the step of translating the legacy instructions and going to said step of executing translated instructions; and otherwise, if legacy instruction data has not been modified, going to said step of executing translated instructions.”

As best understood, instruction is composed of opcode/operator (like Boolean “AND” operator/opcode) and operand (data values for AND operation like X=1 & Y=0).

Art Unit: 2128

The "legacy instruction data" is understood as data part of an instruction, or the operand. Modifying operand does not make an instruction "self-modifying instruction" as understood from common definition as known in the art (also shown above).

Arguments presented are allegations at best. Examiner requests applicant to provide support for the "self modifying code" in claims, as well as in the specification for the presented arguments.

13. Response to Section 12.1

*-- No argument presented other than stating the examiner's previous rejection.*

14. Response to Section 12.2

*Regarding Section 12.2.1-12.2.3:*

Examiner thanks applicant in expressing the teachings of Mann'295 so clearly and concisely with attempts to distinguish the instant invention from teachings of Mann'295. Further, commentary on statistical analysis of data and non-data modification is also noted. However the as stated previously the claims do not recite "self modifying code" and are also not interpreted as "self modifying code", therefore the rejection is maintained.

*Regarding Section 12.2.4:*

Applicant has argued the following:

"In making the rejection as to that claimed element ["storing translation indications..."], however, the Examiner ignores the part of that claimed element that recites "said storing translation indication using a subset of block address digits whereby block numbers in the said table are the same for multiple different blocks,". The examiner ignores this recitation of multiple different blocks in applicant's claim because Mann'295 is one-to-one structure that neither requires nor permits multiple different blocks.

Examiner again respectfully disagrees; as it is shown that having one block or multiple different blocks (which Mann'295) is irrelevant as blocks can start from any instruction (See discussion in Response to 11.2.3.1). Further, Mann'295 clearly teaches reuse of the translated code which implies many (legacy instructions) to one (host code); thereby teaching block numbers in the said table are the same for multiple different blocks. Further, same index-offset information is stored (block address digits) for instructions in the same block (See Mann'295: Fig.3 element 74). Also see Mann'295: Fig.4; Col.9 Lines 10-36, Col.7 Lines 49- Col.8 Line 65.

*Regarding Section 12.2.5:*

The argued portion of the claim states:

12.2.5. Referring specifically to the Examiner's rejection of Claim1 as quoted in Section 12.1 above, the Examiner's statement in Section 12.1.10 "going to the step of executing the translated instruction" is erroneous in that no execution of a "translated instruction" occurs, but rather as is clear from the Examiner's citation (Mann'295: Fig.4 Path 108, 128, 129), processing reverts in Mann'295 to a time-wasting "interpretation" in "Interpret Instruction 128" as shown and described in connection with FIG 4 of Mann '295. At least for this reason, the Examiner's rejection cannot be sustained.

Applicant is correct that "translated instruction" in the second half of the claim limitation is not given weight. The claim limitation reads as follows:

"if the indications indicate that said particular block has not been translated, going to said step of executing translated instructions"

The weight is not given because if the instruction is "not translated" then the how can the step of "executing an translated instruction" can happen. Instead the execution is understood as normal execution, and in view of Mann'295's teachings as interpretation. Please see rejection under 35 USC 112 second paragraph.

*Regarding Section 12.2.6:*

Examiner in view of above interpretation states that current claim does not exclude interpretation and maintains the rejection.

15. Response to Section 12.3-12.6

*Regarding Section 12.3.1-12.3.2:*

Examiner has considered the arguments presented and they seem to restating themselves from section 11.2.3.1. Examiner respectfully refers applicant to response to section 11.2.3.1.

16. Response to Section 12.7

*Regarding Section 12.7.1-12.7.4:*

-- No argument presented other than stating the examiner's previous rejection.

*Regarding Section 12.7.4.1:*

Applicant has argued claim10 requires storing "subset of all the translated blocks" and Mann'295 does not process subsets.

Examiner respectfully disagrees with the allegation as Mann'295 processes legacy instructions in blocks and some of the blocks not including self-modifying code are translated and stored in host code, thereby storing "subset of all the translated blocks" is met (See Mann'295: Fig.4).

*Regarding Section 12.7.4.2-12.7.4.3:*

Limitations regarding storing a count may be allowable. Please see allowable subject matter.

17. Response to Section 12.8

*-- No argument presented other than stating the examiner's previous rejection.*

18. Response to Section 12.9

Same arguments are made for claim 11 with the indicated mapping. Examiner maintains the respective corresponding rejections.

19. Response to Section 12.10

Applicant is arguing is similarly as before that Mann'295 performs wasteful "interpretation" irrespective if the data is modified in the self-modification code, whereas in current invention the a check is made for data modification (In Reference to 12.8.8). Although the applicant's argument is valid, current claim language does not preclude "interpretation".

Further applicant is correct that Mann'295 only checks for self-modifying code, and not specifically data modification (Mann: Col.9 Lines 5-9). However data modification do not make an instruction self-modifying instructions and identification thereof is well known in the art (See US Patents No. 6594734, 5835949) as evidentiary support documents. Examiner maintains the rejection in view of above two reasons.

20. Response to Section 12.11

Applicant has argued the following:

12.11. Additionally, the Examiner's statement in Section 12.8.9 that "Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks" is not supported by the Examiner's citation therefor, "(Mann '295:Col.5 Lines 54-63)", nor by anything else in Mann '295.[1] Rather, the citation (Mann '295:Col.5 Lines 54-63) relied upon by the Examiner indicates that not all blocks are translated, the primary operation being "interpretation". [2] Only after blocks have been repeatedly interpreted a number of times are the blocks translated. Once blocks are translated in Mann '295, they are stored with a one-to-one indexing with no provisions for an subset operations. (See Section 12.2.4 and discussions of subsets in other Sections) [3].

Applicant has first argued storing translation indications for only a subset of all translated blocks, is not supported by Mann'295 [1]. This argument is addressed above in "*Regarding Section 12.7.4.1:*". Further [2] argument seems to on the contrary support the examiner's response. Arguments regarding one-to-one are previously addressed in "*Regarding Section 11.2.3.1*" above.

Response to Section 12.11, 13, 14, and 15

Applicant has traversed the rejection for dependent claims as being dependent from allowable independent claims. Examiner respectfully disagrees and maintains the rejection for dependent claims 12,13,15,16, 17, 19, and 21-27 under Mann'295 in view of Smith'1986 and claims 14 & 28 further in view of Scalzi'013. Claims 6 & 20 remain rejected under Mann'295 in view Scalzi'013.

***Claim Rejections - 35 USC § 112***

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

21. Claims 1-3 and 5-27 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

**Regarding Claim 1, 11, 15 and 25**

These claims disclose executing translated instruction when the instruction is not translated. For example claim1 discloses:

"if the indications indicate that said particular block has not been translated, going to said step of executing translated instructions"

Examiner finds the instruction vague and indefinite. Respective dependent claims remain rejected due to their dependence on the independent claim.

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148

USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).



**22. Claims 1-3, 5, 7-9, 11-13, 15-17, 19, 21-23, 25, and 27 are rejected under 35**

**U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter).**

Regarding Claim 1

**Mann '295** teaches a computer-implemented method of emulating execution of a legacy instruction (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29).

Further, **Mann '295** teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). **Mann '295** teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19).

Further, **Mann '295** teaches storing translations into translation store as the host code block (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28) for each legacy instruction.

Further, **Mann '295** teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).

**Mann '295** teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).

Further, **Mann '295** teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).

Further, **Mann '295** teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).

Further, **Mann '295** teaches if a legacy instruction is not a store instruction, going to step of executing translated instruction as performing the DOCT (Mann '295: Col.9 Lines 5-10; Fig. 4 Elements: 126, 124, Fig. 5 Element 134/136 & 148).

**Mann '295** teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the (block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).

**Mann '295** teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X – do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).

**Mann '295** teaches checking if the instruction has been translated (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the legacy instruction data has been modified, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 –20). If the instruction data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).

**On a closer review**, **Mann'295** also teaches using index/offset (**partial addresses**) from the instructions as index for the block entry table, thereby teaching the limitation of “said storing translation indications using a subset

**of block address digits whereby block numbers in said table are the same for multiple different blocks” (Mann’295: Col. 6 Lines 16-24).**

Mann’295 does not teach the details of the limitation presented above explicitly. Smith ‘1982 teaches that TLB having a hashing mechanism to map the virtual addresses (block numbers) to the real address (translated host code block) (Smith ‘1982: Pg.475 Col.2 Paragraphs 3-4). Hashing (done by taking an XOR or through randomized algorithm) depends on the number of bits selected, resulting in folding or overlapping (Smith ‘1982: Pg.488, Col.1 Paragraph 1; Pg.489, Col.1). The hashing scheme selected by the applicant is extremely simplified version, as only one middle bit is selected to index the translation indication index table.

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Smith ‘1982 and apply them to Mann ‘295 to implement the indexing table as disclosed. The motivation would have been that Smith ‘1982 discloses the necessity for TLB like lookup when dealing with translated information when address space doesn’t map directly (Smith ‘1982: Pg.510, Section 2.9, 2.17). Mann ‘295’s design requires translation as one target code block may have one or more translated host code instructions. Hence Smith ‘1982 solves Mann ‘295’s problem of mapping the target legacy instruction object to translated host object code block (Mann ‘295: Col.6 Lines 47-55). Please see further the response to arguments for clarification.

Regarding Claim 2

Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63).

Regarding Claim 3

Mann '295 storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).

Regarding Claim 5

From the teachings of Mann'295 and Smith'1982 it obvious to use the subset (middle 3 hexadecimal) of digits form the legacy instruction for the part of the address field of the indexing table (Mann'295: Col. 6 Lines 16-24; Smith'1982: Pg.475 Col.2 Paragraphs 3-4; Pg.488, Col.1 Paragraph 1; Pg.489, Col.1).

Regarding Claim 7

Mann '295 teaches that legacy instructions are object code instructions compiled/assembled for the legacy architecture (Mann '295: Col.2 Lines 44-51).

Regarding Claim 8

Mann '295 teaches legacy instruction includes store instructions for modifying instruction code (Mann '295: Col.9 Lines 5-20, 38-47; Col.7 Lines 14-37).

Regarding Claim 9

Mann '295 teaches translation indication includes a state field as tag field, for each block number, indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67).

Regarding Claim 11

**Mann '295** teaches a method of dynamic emulating execution of a legacy instruction (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29). Further, **Mann '295** teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). **Mann '295** teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19). Further, **Mann '295** teaches storing translations into translated code into host code block (translation store) for each legacy instruction (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28).

Further, **Mann '295** teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).

**Mann '295** teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).

Further, **Mann '295** teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).

Further, **Mann '295** teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).

Further, Mann '295 teaches checking store instruction associated to the block entry table, if instruction data is stored (Mann '295: Col.9 Lines 5-10; Fig. 4 Elements: 126,

Art Unit: 2128

124, Fig. 5 Element 134/136 & 148) for a particular block. Mann '295 also teaches checking if the instruction data has been modified (Mann '295: Col.9 Lines 9 –20). Further, Mann '295 teaches bypassing the checking if there is no store instruction data (Mann '295: Fig. 5 Element 134/136 & 148). Further, **Mann '295** teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63).

**Mann '295** teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the (block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).

**Mann '295** teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X – do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).

**Mann '295** teaches checking if the instruction has been translated (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the legacy instruction data has been modified, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 –20). If the instruction data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).

**Mann '295** teaches storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).

**Mann '295** teaches translation indication includes a state field as tag field for each block number indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67).

Regarding Claim 12

Method claim 12 is directed towards similar limitations as the method claim 10 and is rejected for the same reason as claim 10.

Regarding Claim 13

Mann '295 teaches that legacy instructions are object code instructions compiled/assembled for the native legacy architecture executed as guest on host architecture (Mann '295: Col.2 Lines 44-51, Fig.3).

Regarding Claim 15

Mann '295 teaches an apparatus and a method for emulating self-modifying code. The system claim 15 is directed towards the same limitations as the method claim 1 and is rejected for the same reason as claim 1.

Regarding Claim 16

The system claim 16 is directed towards the same limitations as the method claim 2 and is rejected for the same reason as claim 2.

Regarding Claim 17

The system claim 17 is directed towards the same limitations as the method claim 3 and is rejected for the same reason as claim 3.

Regarding Claim 19

The system claim 19 is directed towards the same limitations as the method claim 5 and is rejected for the same reason as claim 5.

Regarding Claim 21

The system claim 21 is directed towards the same limitations as the method claim 7 and is rejected for the same reason as claim 7.

Regarding Claim 22

The system claim 22 is directed towards the same limitations as the method claim 8 and is rejected for the same reason as claim 8.

Regarding Claim 23

The system claim 23 is directed towards the same limitations as the method claim 9 and is rejected for the same reason as claim 9.

Regarding Claim 25

The system claim 25 is directed towards the same limitations as the method claim 11 and is rejected for the same reason as claim 11. Further, Mann '295 storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).

Regarding Claim 27

The system claim 27 is directed towards the same limitations as the method claim 13 and is rejected for the same reason as claim 13.



**23. Claims 6 & 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter) in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).**

Regarding Claim 6

Teachings of Mann '295 are disclosed in the claim 1 rejection above.

Mann '295 does not teach legacy instructions are for a legacy system having S/390 Architecture.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture (Scalzi '013: Col.17 Lines 54-57).

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi '013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

Regarding Claim 20

The system claim 20 is directed towards the same limitations as the method claim 6 and is rejected for the same reason as claim 6.

**24. Claims 14 & 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter) in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter), further in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).**

Regarding Claim 14

Teachings of Mann '295 & Smith '1982 are disclosed in the claim 11 rejection above.

Mann '295 & Smith '1982 do not teach emulated execution by translation from CISC based legacy instruction set to the RISC based target/host instruction set.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture which is a CISC architecture and host architecture is power PC (RISC) based architecture (Scalzi '013: Col.17 Lines 54-57).

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi '013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

Regarding Claim 28

Teachings of Mann '295 & Smith '1982 are disclosed in the claim 25 rejection above.

The system claim 28 is directed towards the same limitations as the method claim 14 and is rejected for the same reason as claim 14. To facilitate prosecution rejection is repeated below.

Mann '295 & Smith '1982 do not teach emulated execution by translation from CISC based legacy instruction set to the RISC based target/host instruction set.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture which is a CISC architecture and host architecture is power PC, RISC based architecture (Scalzi '013: Col.17 Lines 54-57).

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi '013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

***Allowable Subject Matter***

25. Claims 10, 24 and 26 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.
26. The following is a statement of reasons for the indication of allowable subject matter:
- Mann '295 does not teaches incrementing/decrementing the count for the modified/removed block, but teaches equivalent functionality of removing and de-allocating the block entry table (indexing table entry) when the translated host block is no longer needed/garbage collected (Mann '295: Col.7 Lines 62-67, Col.8 Lines 1-3). Mann '295 also teaches that the translation indication includes a state field (tag field), for each block number, indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67). Further, Mann '295 teaches incrementing the state field each time block is executed on the cache (Mann '295: Col.6 Lines 64-65). Therefore the count is not associated to the addition or removal of block.

***Conclusion***

27. All claims are rejected.

28. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

29. **Examiner's Note:** Examiner has cited particular columns and line numbers in the references applied to the claims above for the convenience of the applicant.

Although the specified citations are representative of the teachings of the art and are applied to specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested from the applicant in preparing responses, to fully consider the references in their entirety as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the Examiner.

In the case of amending the claimed invention, Applicant is respectfully requested to indicate the portion(s) of the specification which dictate(s) the structure relied on for proper interpretation and also to verify and ascertain the metes and bounds of the claimed invention.


***Communication***

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Akash Saxena whose telephone number is (571) 272-8351. The examiner can normally be reached on 9:30 - 6:00 PM M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kamini S. Shah can be reached on (571)272-2279. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Akash Saxena  
Patent Examiner, GAU 2128  
(571) 272-8351  
Monday, January 08, 2007

  
Kamini S. Shah  
Supervisory Patent Examiner, GAU 2128  
Structural Design, Modeling, Simulation and Emulation